How to Make Use of Empirical Knowledge about Tester's Information Needs

Anne Hess, Joerg Doerr
Fraunhofer IESE
Kaiserslautern, Germany
{anne.hess, joerg.doerr} @iese.fraunhofer.de

Norbert Seyff
University of Applied Sciences and Arts Northwestern Switzerland and University of Zurich, Switzerland
norbert.seyff@fhnw.ch

***Abstract*— Software requirements specifications (SRS) serve as a source of communication and information for a variety of roles involved in development activities. From the viewpoint of these SRS consumers, which includes testers as one of the key customers, the analysis of requirements specifications is often frustrating as it is time consuming and often requiring a lot of cognitive effort due to the increasing complexity of the documented information. Filtering the large amount of information by generating views that fit role-specific demands of SRS consumers is a promising solution approach for tackling this problem. This paper discusses concepts and key functionalities of an initial tool implementation of our proposed solution that is based on detailed knowledge about information needs that we gained in a series of empirical studies. Furthermore, we present potential usage scenarios illustrating its application in industry from the viewpoint of a tester.**

***Index Terms*—requirements specification, role-specific views, tool, usage scenarios, testers**

I. Introduction

Software engineering (SE) projects are inherently cooperative and require software engineers (who may have specific roles) to exchange information and coordinate their efforts [1]. A shared understanding regarding the requirements of stakeholders that are to be supported by a software system is therefore required. Requirements engineering (RE) supports the specification of requirements with the help of different requirements artifacts. These artifacts can include information both in the form of natural language and conceptual models using different notations [2]. For example, detailed information about supported stakeholders can be specified using textual role descriptions [2] or personas [3]; information about goals can be specified using goal models ([2][4][5]); and interactions between the system and actors in its environment can be specified by means of textual specifications of use cases [2].

All these artifacts are typically consolidated, structured, and maintained in software requirements specifications (SRS), which often comprise a very large number of such artifacts due to the increasing complexity of software systems.

Such complex SRS serve as an important source of knowledge to a variety of SRS consumers – in the following referred to as "artifact stakeholders" – who are involved in subsequent SE activities like testing or architecture design.

Thus, the artifact stakeholders such as testers need to actively work with the SRS and continuously analyze the documented artifacts in order to adequately perform their role-specific tasks e.g., plan, prepare and run system tests based on the SRS.

Results obtained by interviews, analyzing industrial RE practices [6] and by assessing industrial projects with the RE Assessment Guide [7] have revealed that the usage and analysis of SRS is often difficult and time-consuming for artifact stakeholders. Besides the aforementioned complexity, this observation might be attributed to insufficient quality of the artifacts [8] and particularly to the fact that the creation of SRS tailored to role-specific information needs is not sufficiently supported by current RE approaches and tools [9]

That means that from the viewpoint of a particular artifact stakeholder like a tester, important requirements artifacts might be spread over different sections in a given SRS, be delivered too late, be specified on an inappropriate level of detail, or even be missing. Or the SRS might include a variety of requirements artifacts that are not important for accomplishing particular test-specific tasks such as preparing and running system tests [9].

A promising approach for handling the aforementioned problems seems to be a solution that provides each artifact stakeholder with predefined views on a given SRS that fit their specific information demands [9]. We claim that such a solution is highly relevant for industry - specifically for teams implementing large-scale projects where there are a large number of roles involved, each with unique information needs.

In this context, an insufficient satisfaction of these information needs is critical: It may lead to delays and frustration in subsequent SE activities, which could lead to disregard or ignorance of SRS by artifact stakeholders and ultimately to costly changes, expectation failures, budget or time overruns [9].

The remainder of paper is structured as follows: In Section II, we introduce concepts and key functionalities of our current tool to provide the specific views. Section III introduces four usage scenarios that reflect typical tool applications in industry from the viewpoint of a tester. The paper concludes with a summary and outlook on future work in section IV.

II. Tool Concept and Implementation

We realized our initial solution [15] as extensions of Microsoft Excel®, a common tool that is often used in practical settings in order to create SRS. All implemented functionalities (macros) have been encapsulated as add-ins that can easily be imported and activated in any Microsoft Excel® application.

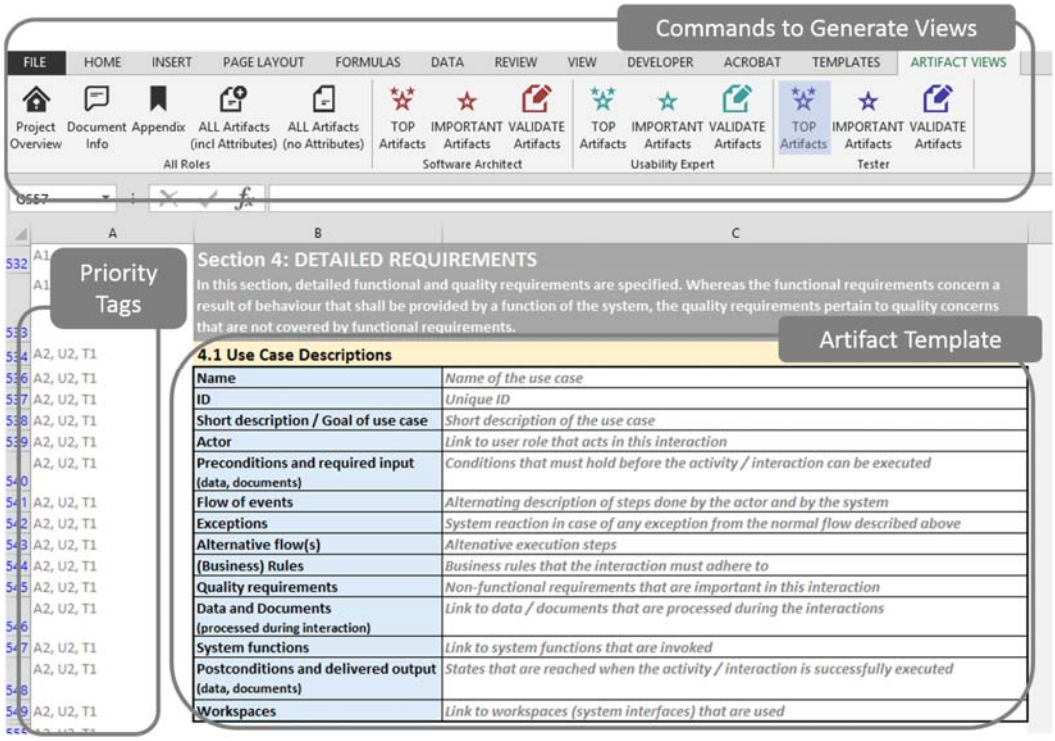

Fig. 1. Excerpt of SRS Template Visualizing a Use Case Template (as an example Artifact Template), Priority Tags (Column A). and Commands for Generating Role-specific Views (Menu)

A. Background

To generate suitable views that fit role-specific demands, we claim that detailed, empirically valid knowledge about role-specific information needs is required first. In previously published work [9] [11], we discussed first lessons learned and initial results regarding such role-specific information needs. Since then we have further supplemented and refined these initial results by conducting a series of empirical studies that investigated priorities of a set of requirements artifacts from the viewpoint of testers, software architects and usability engineers.

Please note that a further and detailed discussion of the empirical work is not in the scope of this paper. However, we would like to discuss the requirements artefacts which were the basis for our investigation.

We followed the task-oriented RE approach (TORE) [12] that proposes the following requirements artifacts:

Descriptions of stakeholders that capture relevant information and characteristics about stakeholders who are to be supported by or have an influence on the system to be built.

Descriptions of project goals / stakeholder goals that are to be fulfilled by the system to be built. These goal descriptions typically include and refine the vision of the system.

Descriptions of as-is situations that illustrate current situations without the system to be built (e.g., current problems that motivate the need for a new system; how supported business processes are currently being performed).

Descriptions of to-be situations that illustrate the situation in the future with the system (e.g., how supported business processes will be performed with the system to be built).

Descriptions of the system context that define the system's environment (e.g., users, external systems) including an overview on functionalities that the system offers to it.

Descriptions of interactions that describe how the system interacts with entities in its environment (e.g., users).

Descriptions of system functions that specify input, internal behavior, and output of system functionalities.

Descriptions of quality requirements that specify desired qualities (non-functional requirements) of the system to be built (e.g., regarding performance, availability, dependability).

Descriptions of technical constraints that limit the solution space beyond what is necessary for meeting the requirements.

B. Tool Solution Concepts

The overall tool solution concept comprises an *SRS template including artifact templates* as well as *priority tags & filter rules* that define the various role-specific views. These concepts are described in more detail in the following.

1) *Templates:* The *SRS template* structures the SRS document into different (sub)sections, as it is done in standards such as IEEE 830-1998 [10].

In order to specify requirements artifacts in detail, predefined *artifact templates* are provided in each of the subsections of the SRS template. These artifact templates represent the TORE requirements artifacts that have been investigated in the empirical studies (see Section II.A).

Figure 1 illustrates an example artifact template which supports the specification of interactions between the system and its actors in the form of textual use case descriptions.

2) *Priority Tags and Filter Rules:* Each artifact template and its description attributes are tagged with so-called "priority tags" (see column A in Figure 1). In the case of the example, the artifact template is tagged with A2, U2, and T1. These priority tags correspond with the priority of a particular requirements artifact from the viewpoint of software architects (A), usability engineers (U), and testers (T) that we calculated based on the empirical data gained.

There are three classifications of artifacts and corresponding priority tags for each of the three perspectives:

High priority artifacts include key information for the artifact stakeholders that is critical to fulfill their role-specific tasks. These artifacts have to be specified timely and precisely. This classification is represented by the tags A1, U1, and T1 for the three artifact stakeholder groups.

Lower priority artifacts include relevant information for the artifact stakeholders that is, in contrast to high priority artifacts, less critical. That is, the artifact stakeholders could also do their tasks based on high-level descriptions of these artifacts (i.e., diagrams without additional textual details). This classification is represented by the tags A2, U2, and T2 for the three artifact stakeholder groups.

Unimportant artifacts do not include relevant information for the artifact stakeholders. This classification is represented by the tags A3, U3, and T3.

The descriptions of use cases for example (see Figure 1), is of high priority for testers (T1) and of lower priority for software architects and usability engineers (A2, U2).

Based on the priority tags, a set of *filter rules* was implemented to finally generate role-specific views. In our tool, we used the filter functionalities provided by Microsoft Excel® that allow reducing the number of displayed rows in a sheet based on certain cell values in a given column.

To define the filter rules for the views, we used the values of the priority tags in column A (see Figure 1). For example, the execution of the filter rule "Filter based on T1" enables that the number of displayed rows in the SRS is reduced to the number of rows that contain the value (priority tag) "T1" in column A.

Thus, the execution of this filter rule creates a view that displays only artifacts that are of high priority for testers (T1).

C. Role-Specific Views

According to the aforementioned scheme, we implemented various filter rules in form of macros that can be executed via the different menu items in the menu *(*Figure 1*)*. The tool offers the following role-specific views:

TOP Artifacts: displays only artifacts that are of high priority for the corresponding role (i.e., artifacts that are tagged with A1, U1, and T1 respectively)

IMPORTANT Artifacts: displays artifacts that are of lower priority but still important to the corresponding role (i.e., artifacts tagged with A2, U2, and T2)

VALIDATE Artifacts: This view is intended to support validation activities of the SRS for the purpose of quality assurance following e.g. perspective-based techniques like [2]. Thus, this view displays all artifacts that belong to both the view TOP Artifacts and IMPORTANT Artifacts (i.e., artifacts tagged with A1 or A2, U1 or U2, and T1 or T2).

Besides the aforementioned priority-based views, our current implementation also offers additional views to the users (see menu bar in Figure 1) that are intended to reduce the complexity of displayed information in the SRS. These views were also realized in the form of filter rules based on predefined tags in column A and comprise:

Project Overview: displays artifacts that allow getting a good overview of the project. This includes general project descriptions (customer, timeline, budget), motivation, stakeholders, vision, as well as high-level descriptions of the system context (such as use case diagrams [2]).

Document Info: displays meta-information about the SRS. This includes authors, version number, change history, etc.

Appendix: displays information specified in the Appendix of the SRS. This includes artifacts such as glossaries.

ALL Artifacts (incl. Attributes): displays all requirements artifacts including meta-information such as author, version, source, cross-references, and validation status.

ALL Artifacts (no Attributes): this view displays all requirements artifacts but without meta-information as it might not always be relevant for the artifact stakeholders. Thus, this alterative might be helpful to additionally handle the complexity of information specified in the SRS.

III. USAGE SCENARIOS

In the following, we present three scenarios that illustrate envisioned applications of our proposed solution in industrial project settings, particularly from the perspective of testers.

Usage Scenario 1: Reduce the complexity of SRS to support (continuous) document analysis. Once a stable version of requirements artifacts has been documented in the SRS, the testers typically analyze the documented artifacts to perform their role-specific tasks. Thus, a tester might first get an overview of the project (via view Project Overview) which provides him with a condensed summary of project goals, relevant stakeholders and the system scope. Afterwards he might analyze the most important artifacts for performing his main tasks (preparing and running system tests) via the view TOP Artifacts that - based on the empirical data - comprises detailed descriptions of interactions in form of use cases (see template in Figure 1). During planning activities, he might also refer to further important information via the view IMPORTANT Artifacts which comprises e.g., information helping to understand the relation between use cases and system functions. During this document analysis activities, (rather) unimportant information for the tester (according to the empirical data, e.g., stakeholder characteristics) is hidden but can be accessed anytime via the views ALL Artifacts or Project Overview.

We conclude that with views the typically large amount of specified information can be filtered in order to reduce time and cognitive effort that might otherwise be spent on searching.

The presented scenario assumes that a waterfall-like approach is applied i.e., the SRS is completed before subsequent development starts. However, also iterative software development processes (e.g., RUP [13]) can be supported by

views. For example, a tester might have a look at initial drafts of use cases in order to plan testing activities. Later on, he might revisit the updated use cases to specify detailed test cases.

Thus, the testers can also benefit in such continuous document analysis activities, as they do not need to browse through the complete SRS every time they want to check for an update. Instead, they can focus on updates of important artifacts.

Usage Scenario 2: Align the elicitation and documentation of requirements artifacts with role-specific needs. With the help of the various views, requirements engineers can align their elicitation and documentation activities with role-specific needs.

This could be beneficial in iterative project settings where on-time delivery of important information for subsequent development activities is critical for project success [7]. Also DevOps that positively effects quality assurance performance could benefit as this framework is supported by a culture of collaboration, automation, and information sharing [14].

Usage Scenario 3: Improve communication and quality of SRS. Also the collaboration and communication between requirements engineers and testers can benefit from our solution. The views – particularly the underlying knowledge about role-specific needs - could enable the requirements engineer to directly contact and include testers already during elicitation and documentation of artifacts that are (highly) relevant for testers.

This could enhance communication in the project and improve the quality of the SRS already during its creation. This might also reduce solving open issues between testers regarding artifacts after they are delivered by the requirements engineer.

Finally, once an initial SRS version has been created, the requirements engineer may initiate review activities of the SRS (e.g., following perspective-based techniques [2]). The tester could generate a selective role-specific view (via VALIDATE Artifacts) and review the artifacts contained in this view according to a predefined review process. The review might be more efficient, as unimportant information for a tester is automatically hidden and does not have to be reviewed.

IV. Summary and Future Work

With the presented work we have introduced practical applications of a tool that generates role-specific views on a given SRS based on empirical data [15]. We claim that such a solution has the potential to support especially large development teams implementing large-scale projects to handle the complexity of SRS in an efficient manner.

We implemented an initial tool version as extensions of Microsoft Excel® with the overall goal of using this tool in case studies and experiments to gain insights about possible benefits and limitations of our solution idea. Furthermore, is would also be possible to apply our solution in requirements management tools that provide filter options based on attributes. However, we consider the underlying empirical knowledge about role-specific information needs as a major contribution to industry.

An initial case study has revealed that role-specific views have the potential to support both artifact stakeholders and requirements engineers in typical usage scenarios as described in this paper. To mitigate current limitations (such as students as participants, limited system scope with limited numbers of artifacts) and increase the validity of these initial results, we will continue with further evaluations in industrial settings. This will also include research on factors that might influence information needs (such as background, personalities) and how to enable artifact stakeholders to stay "in control" and not have the feeling that they are missing information if the tool hides information that they believe might be important for them.

References

[1] J. Whitehead, "Collaboration in software engineering: a roadmap," Future of Software Engineering, 2007, FOSE '07, pp.214–225, 23-25 May 2007.

[2] K. Pohl and C. Rupp, Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant. Rocky Nook, Inc., 2011.

[3] A. Cooper, The Inmates are Running the Asylum. Macmillan Publishing Co., Inc., Indianapolis, USA, 1999.

[4] E. Yu, P. Giorgini, N. Maiden, and J. Mylopoulos, Social Modeling for Requirements Engineering. The MIT Press, 2011.

[5] A. Van Lamsweerde and E. Letier. "From object orientation to goal orientation: a paradigm shift for requirements engineering," Radical Innovations of Software and Systems Engineering in the Future. Springer Berlin Heidelberg, 2004, pp.325–340.

[6] S. Adam, J. Doerr, and M. Eisenbarth, "Lessons learned from best practice-oriented process improvement in requirements engineering – a glance into current industrial RE application," REET09, 2009.

[7] D. Rapp, A. Hess, N. Seyff, P. Sporri, E. Fuchs, and M. Glinz, "Lightweight requirements engineering assessments in software projects", In Proc. 22nd IEEE Int. Requirements Engineering Conference (RE), 2014, pp.354–363, 25-29 Aug. 2014.

[8] H. Femmer, J. Mund, and D. Méndez Fernández, "It's the activities, stupid!: a new perspective on RE quality", 2nd International Workshop on Requirements Engineering and Testing (RET '15), 2015, pp. 13-19.

[9] A. Gross[1], J. Doerr, "What you need is what you get!: the vision of view-based requirements specifications," In Proc. 20th IEEE International Requirements Engineering Conference (RE), 2012, pp.171–180, 24-28 Sept. 2012. ([1] same author as A. Hess)

[10] IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document. IEEE Standard 1362-1998, 1998.

[11] A. Gross[2] and J. Doerr, "What do software architects expect from requirements specifications? results of initial explorative studies," 1st International Workshop on the Twin Peaks of Requirements and Architecture", 2012, pp.41–45 ([2] same author as A. Hess).

[12] S. Adam, N. Riegel, J. Doerr, "TORE - a framework for systematic requirements development in information systems," Requirements Engineering Magazine, Issue 2014 – 04, http://re-magazine.ireb.org/issues/2014-4-steady-flight/tore/ (last access June 19th 2017).

[13] P. Kruchten The rational unified process: an introduction. 2nd edition. Addison-Wesley, Boston, 2000.

[14] F. Erich, C. Amrit, M. Daneva, "A mapping study on cooperation between information system development and operations," PROFES 2014. LNCS, vol. 8892, 2014, pp. 277–280.

[15] Downloadlink https://oc.iese.de/index.php/s/21Rl1jntvE9oZQu